IN THE CLAIMS

1. (Currently amended) A method comprising:

allocating a first address space for a guest operating system;

allocating a second address space for a virtual machine monitor (VMM);

mapping a first portion of the VMM into the first address space and the second address space;

locating a second portion of the VMM in the second address space;

detecting that [[a]] the guest operating system attempts to access a region occupied by [[a]] the first portion of ~~a virtual machine monitor (~~ the VMM[[)]] within [[a]] the first address space; and

relocating the first portion of the VMM within the first address space to allow the guest operating system to access the region previously occupied by the first portion of the VMM.

2. (Original) The method of claim 1 wherein the first portion of the VMM includes a set of VMM code and data structures that are architecturally required to reside in the first address space.

3. (Original) The method of claim 1 wherein the first portion of the VMM includes a set of trap handlers and an interrupt-descriptor table (IDT).

4. (Currently amended) The method of claim 1 further comprising:

dividing the VMM into the first portion and a second portion~~;~~

~~creating the first address space associated with the guest operating system;~~

~~creating a second address space associated with the VMM;~~

~~locating the second portion of the VMM in the second address space associated with the VMM; and~~

~~mapping the first portion of the VMM into the first address space and the second address space~~.

5. (Original) The method of claim 1 further comprising:

receiving control over an event initiated by the guest operating system when the event may potentially cause an address space conflict between the guest operating system and the VMM.

6. (Original) The method of claim 5 wherein receiving control further comprises:

setting access rights of the section occupied by the first portion of the VMM to a more privileged level than a privilege level associated with the guest operating system; and

receiving a trap caused by an attempt of the guest operating system to access a hardware resource having a higher privilege level than the privilege level associated with the guest operating system.

7. (Original) The method of claim 6 further comprising:

determining that the trap can be handled by the first portion of the VMM;

executing code associated with the trap; and

returning control over the event to the guest operating system.

8. (Original) The method of claim 6 further comprising:

determining that the trap should be handled by the second portion of the VMM;

delivering the trap to the second portion of the VMM;

passing control over the event to the guest operating system after code associated with the trap was executed by the second portion of the VMM.

9. (Original) The method of claim 1 wherein relocating the first portion of the VMM further comprises:

finding an unused region within the first address space; and

re-mapping the first portion of the VMM into the unused region.

10. (Original) The method of claim 1 wherein relocating the first portion of the VMM further comprises:

determining that no unused region exists within the first address space;

selecting a random region within the first address space;

copying content of a memory located at the random region to the second address space; and

re-mapping the first portion of the VMM into the random region.

11. (Original) The method of claim 10 further comprising:

receiving control over an event initiated by the guest operating system, the event corresponding to an attempt of the guest operating system to access the content of the memory previously located at the random region; and

accessing the copied content of the memory in the second address space.

12. (Original) The method of claim 11 further comprising periodically relocating the first portion of the VMM to random regions within the first address space until finding a region that is infrequently accessed.

13. (Currently amended) An apparatus comprising:

a first address space associated with a guest operating system;

a second address space associated with a virtual machine monitor (VMM); and

a virtual machine kernel to allocate the first address space for the guest operating system, to allocate the second address space for the VMM, to map a first portion of the VMM into the first address space and the second address space, to locate a second portion of the VMM in the second address space, to detect that the guest operating system attempts to access a region occupied by [[a]] the first portion of the VMM within the first address space and to relocate the first portion of the VMM within the first

address space to allow the guest operating system to access the region previously occupied by the first portion of the VMM.

14. (Original) The apparatus of claim 13 wherein the first portion of the VMM includes a set of VMM code and data structures that are architecturally required to reside in the first address space.

15. (Original) The apparatus of claim 13 wherein the first portion of the VMM includes a set of trap handlers and an interrupt-descriptor table (IDT).

16. (Currently amended) The apparatus of claim 13 wherein the virtual machine kernel is to divide the VMM into the first portion and the second portion~~, to locate the second portion of the VMM in the second address space associated with the VMM, and to map the first portion of the VMM into the first address space and the second address space~~.

17. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to receive control over an event initiated by the guest operating system when the event may potentially cause an address space conflict between the guest operating system and the VMM.

18. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to receive control by setting access rights of the section occupied by the first portion of the VMM to a more privileged level than a privilege level associated with the guest operating system, and by receiving a trap caused by an attempt of the guest operating system to access a hardware resource having a higher privilege level than the privilege level associated with the guest operating system.

19. (Original) The apparatus of claim 18 wherein the virtual machine kernel is to further determine that the trap can be handled by the first portion of the VMM, to execute code associated with the trap, and to return control over the event to the guest operating system.

20. (Original) The apparatus of claim 18 wherein the virtual machine kernel is to further determine that the trap should to handled by the second portion of the VMM, to deliver the trap to the second portion of the VMM, and to pass control over the event to the guest operating system after code associated with the trap was executed by the second portion of the VMM.

21. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to relocate the first portion of the VMM by finding an unused region within the first address space and re-mapping the first portion of the VMM into the unused region.

22. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to relocate the first portion of the VMM by determining that no unused region exists within the first address space, selecting a random region within the first address space, copying content of a memory located at the random region to the second address space, and re-mapping the first portion of the VMM into the random region.

23. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to receive control over an event initiated by the guest operating system, the event corresponding to an attempt of the guest operating system to access the content of the memory previously located at the random region, and to access the copied content of the memory in the second address space.

24. (Original) The apparatus of claim 13 wherein the virtual machine kernel is to periodically relocate the first portion of the VMM to random regions within the first address space until finding a region that is infrequently accessed.

25. (Currently amended) A system comprising:

a memory to include a first address space associated with a guest operating system and a second address space associated with a virtual machine monitor (VMM); and

a processor, coupled to the memory, to allocate the first address space for the guest operating system, to allocate the second address space for the VMM, to map a

first portion of the VMM into the first address space and the second address space, to locate a second portion of the VMM in the second address space, to detect that the guest operating system attempts to access a region occupied by [[a]] the first portion of the VMM within the first address space and to relocate the first portion of the VMM within the first address space to allow the guest operating system to access the region previously occupied by the first portion of the VMM.

26. (Original) The system of claim 25 wherein the first portion of the VMM includes a set of VMM code and data structures that are architecturally required to reside in the first address space.

27. (Original) The system of claim 25 wherein the first portion of the VMM includes a set of trap handlers and an interrupt-descriptor table (IDT).

28. (Currently amended) A computer readable medium that provides instructions, which when executed on a processor, cause said processor to perform operations comprising:

allocating a first address space for a guest operating system;

allocating a second address space for a virtual machine monitor (VMM);

mapping a first portion of the VMM into the first address space and the second address space;

locating a second portion of the VMM in the second address space;

detecting that [[a]] the guest operating system attempts to access a region occupied by [[a]] the first portion of ~~a virtual machine monitor (~~ the VMM~~)~~ within [[a]] the first address space; and

relocating the first portion of the VMM within the first address space to allow the guest operating system to access the region previously occupied by the first portion of the VMM.

29. (Original) The computer readable medium of claim 28 comprising further instructions causing the processor to perform operations comprising:

finding an unused region within the first address space; and

re-mapping the first portion of the VMM into the unused region.

30. (Original) The computer readable medium of claim 28 comprising further instructions causing the processor to perform operations comprising:

determining that no unused region exists within the first address space;

selecting a random region within the first address space;

copying content of a memory located at the random region to the second address space; and

re-mapping the first portion of the VMM into the random region.